DATA WAREHOUSE AND DATA MINING

Innovative Project Topic:
Detection of fake and phishing websites using Data Mining Techniques

FINAL PROJECT REPORT

Submitted By:

Anukriti (2K17/IT/027)
Anurag Mudgil (2K17/IT/029)

PROBLEM STATEMENT

A phishing website is a spoofed site which often appears as an exact replica of a legitimate site to the users, but it is actually a front which tricks users into providing password credentials or other sensitive information to a malicious cyber actor. It's one of the oldest types of cyberattacks, dating back to the 1990s, and it's still one of the most widespread and pernicious, with phishing messages and techniques becoming increasingly sophisticated. Nearly a third of all breaches in the past year involved phishing, according to the 2019 Verizon Data Breach Investigations Report.

Severity of this problem:

- Perhaps one of the most consequential phishing attacks in history happened in 2016, when hackers managed to get Hillary Clinton campaign chair John Podesta to offer up his Gmail password.
- The "fappening" attack, in which intimate photos of a number of celebrities were made public, was originally thought to be a result of insecurity on Apple's iCloud servers, but was in fact the product of a number of successful phishing attempts.
- In 2016, employees at the University of Kansas responded to a phishing email and handed over access to their paycheck deposit information, resulting in them losing pay.

The phishing attacks taking place today are sophisticated and increasingly more difficult to spot. A study conducted by Intel found that 97% of security experts fail at identifying phishing emails from genuine emails. There is hardly a week when you go to Google News and don't find a news article about Phishing. In the past year, hackers were sending phishing emails to Disney+ subscribers, 'Shark Tank' star Barbara Corcoran lost almost $400K in a phishing scam, a bank issues phishing warnings, and almost three-quarter of all phishing websites now use SSL. For cyber-espionage attacks, that number jumps to 78%. The worst phishing news for 2020 is that its perpetrators are getting much, much better at it thanks to well-produced, off-the-shelf tools and templates.

Detection of phishing websites is a really important safety measure for most of the online platforms. In order to save a platform with malicious requests from such websites, it is important to have a robust phishing detection system in place. Hence, in this report, we present a web application by which a user can easily detect a phishing website and for this, we will be describing the entire methodology step by step along with various decisions made during this process.

RELATED WORK

Phishing is a cyber-attack which targets naive online users tricking into revealing sensitive information. They can at times cause a high damage so it is very important to detect these malicious websites. Detection of phishing websites using an efficient feature-based machine learning framework paper by Routhu Srinivasa Rao & Alwyn Roshan Pais detect phishing websites using 8 different machine learning techniques and later they deduce the Random Forest (RF) algorithm performed the best with an accuracy of 99.31%. Then they performed the experiment with different Random Forests and deduced Random Forest (PCA-RF) performed the best out of all oblique Random Forests (oRFs) with an accuracy of 99.55%. Detecting Phishing Websites Using Machine Learning by Amani Alswailem & Bashayr Alabdullah develops an intelligent system for detecting phishing websites. The system acts as an additional functionality to an internet browser as an extension that automatically notifies the user when it detects a phishing website. They also used Random Forests with accuracy of 98.8% and combination of 26 features. Predicting Phishing URL Using Classification Mining Techniques with Experimental Case Studies - IEEE Conference Publication This paper provides a novel approach to overcome the difficulty and complexity in detecting. It uses an intelligent resilient and effective model that is based on using association and classification Data Mining algorithms. Six different classification algorithms and techniques to extract the phishing training data sets criteria to classify their legitimacy were implemented. Phishing URL Detection with ML is a blog that explains phishing domain (or Fraudulent Domain) characteristics, the features that distinguish them from legitimate domains, why it is important to detect these domains, and how they can be detected using machine learning and natural language processing techniques. It highlights the decision tree methods for detecting fake URLs and websites. Phishing URL Detection Using URL Ranking — Texas Tech University Scholars is a conference paper classifies URLs automatically based on their lexical and host-based features.Online URL reputation services are used in order to categorize URLs and the categories returned are used as a supplemental source of information that would enable the system to rank URLs. The classifier achieves 93-98% accuracy by detecting a large number of phishing hosts, while maintaining a modest false positive rate. URL clustering, URL classification, and URL categorization mechanisms work in conjunction to give URLs a rank.Malicious URL Detection using Machine Learning: A Survey provides a comprehensive survey and a structural understanding of Malicious URL Detection techniques using machine learning. We present the formal formulation of Malicious URL Detection as a machine learning task, and categorize and review the contributions of literature studies that addresses different dimensions of this problem.

OUR METHODOLOGY

Uniform Resource Locator (URL) is created to address web pages. The attacker can register any domain name that has not been registered before. This part of the URL can be set only once. The phisher can change FreeURL at any time to create a new URL.

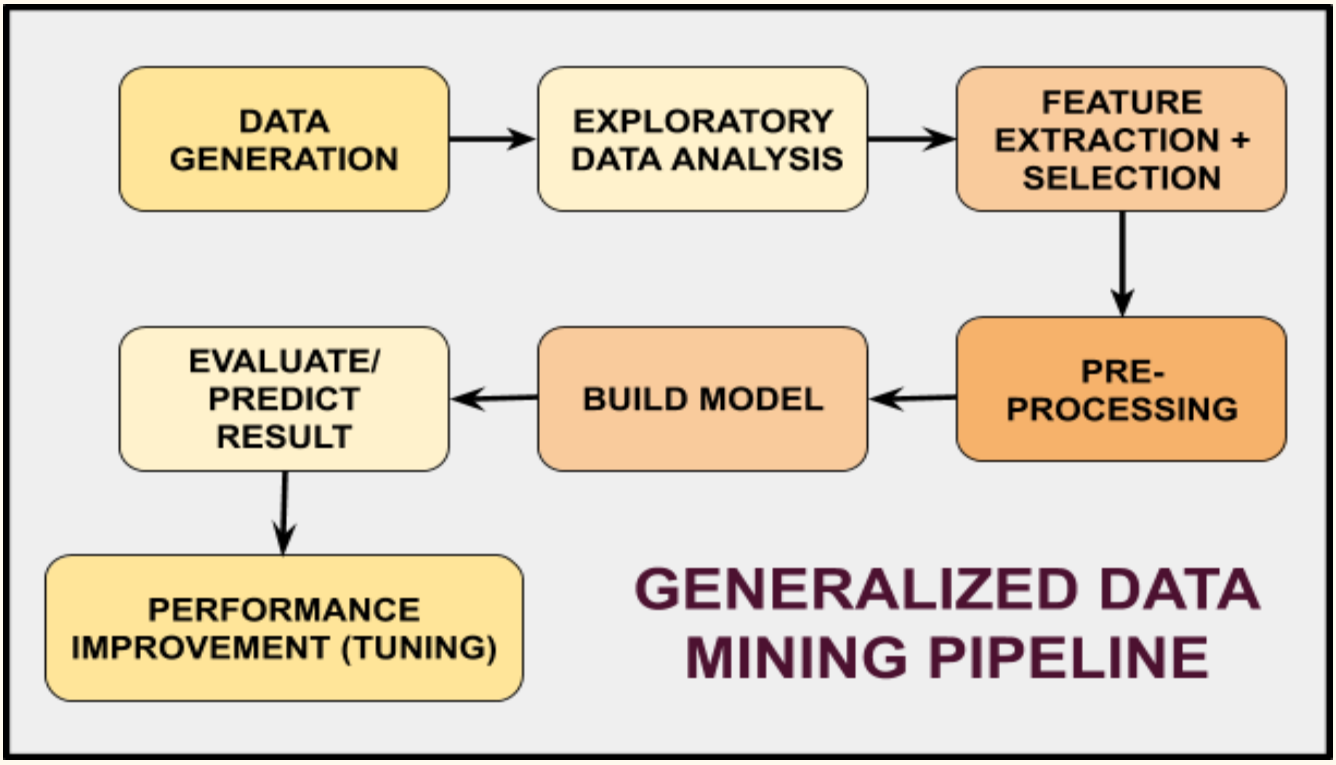
The reason security defenders struggle to detect phishing domains is because of the unique part of the website domain (the FreeURL). These malicious URLs are inserted within the body of the message and are designed to make it appear that they go to the spoofed organization using that organization's logos and other legitimate contents. URL is the first thing to analyse a website to decide whether it is a phishing or not. As we mentioned before, URLs of phishing domains have some distinctive points. Features which are related to these points are obtained when the URL is processed.

Some of URL-Based Features are given below:

- Digit count in the URL
- Total length of URL
- Checking whether the URL is Typosquatting or not. (google.com → goggle.com)
- Checking whether it includes a legitimate brand name or not (apple-icloud-login.com)
- Number of subdomains in URL

Using these features we can train our model using labeled data which has samples whose classes are precisely known as phish domains and legitimate domains in the training phase and predict whether a given URL is fake or genuine.

This project follows a general pipeline as shown below:



STEP 1 (Defining Problem Statement):

We can consider this problem of detecting phishing websites as a binary classification task with 2 possible data labels: Phishing and normal website. This is a typical supervised learning task to predict into either of the two labels given the features.

STEP 2 (Relevant Data Collection):

It was important to find a relevant training dataset and luckily, we found this dataset- UCI phishing websites.

This dataset was well researched and benchmarked by the relevant community of researchers and the accompanying wiki of the dataset comes with a data description document also which discusses the data generation strategies taken by the authors of the dataset. This document highlights features and considerations by the author and it can be found here.

Description of the dataset:

The dataset was collected by analyzing a collection of 2456 websites among which some were used for phishing and others not. For each website included in the dataset, **30 attributes** are given: having_IP_Address {-1,1}, URL_Length {1,0,-1}, Shortining_Service {1,-1}, having_At_Symbol {1,-1}, double_slash_redirecting {-1,1}, Prefix_Suffix {-1,1}, having_Sub_Domain {-1,0,1}, SSLfinal_State {-1,1,0}, Domain_registeration_length {-1,1}, Favicon {1,-1}, port {1,-1}, HTTPS_token {-1,1}, Request_URL {1,-1}, URL_of_Anchor {-1,0,1}, Links_in_tags {1,-1,0}, SFH {-1,1,0}, Abnormal_URL {-1,1}, Submitting_to_email {-1,1}, Redirect {0,1}, on_mouseover {1,-1}, RightClick {1,-1}, popUpWidnow {1,-1}, Iframe {1,-1}, age_of_domain {-1,1}, DNSRecord {-1,1}, web_traffic {-1,0,1}, Page_Rank {-1,1}, Google_Index {1,-1}, Links_pointing_to_page {1,0,-1} and Statistical_report {-1,1}. Significance of each of these features can be studied with the help of this research paper.

Result {-1,1}: Each website in the dataset is labeled by -1 if it is a phishing website and by 1 if it is a normal/benign website.

STEP 3 (Exploratory Data Analysis):

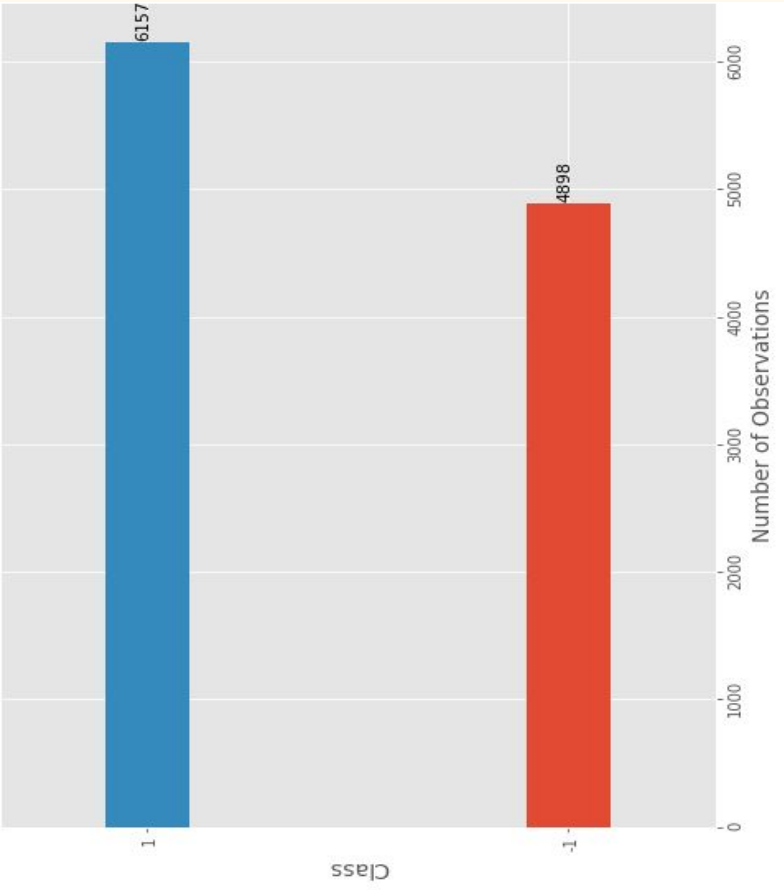
General observations and checking missing values:
This information is shown below:

Number of variables	31
Number of observations	11055
Missing cells	0
Missing cells (%)	0.0%

Checking nature of variables: There are 30 categorical variables as per the analysis as shown:

CAT	30
BOOL	1

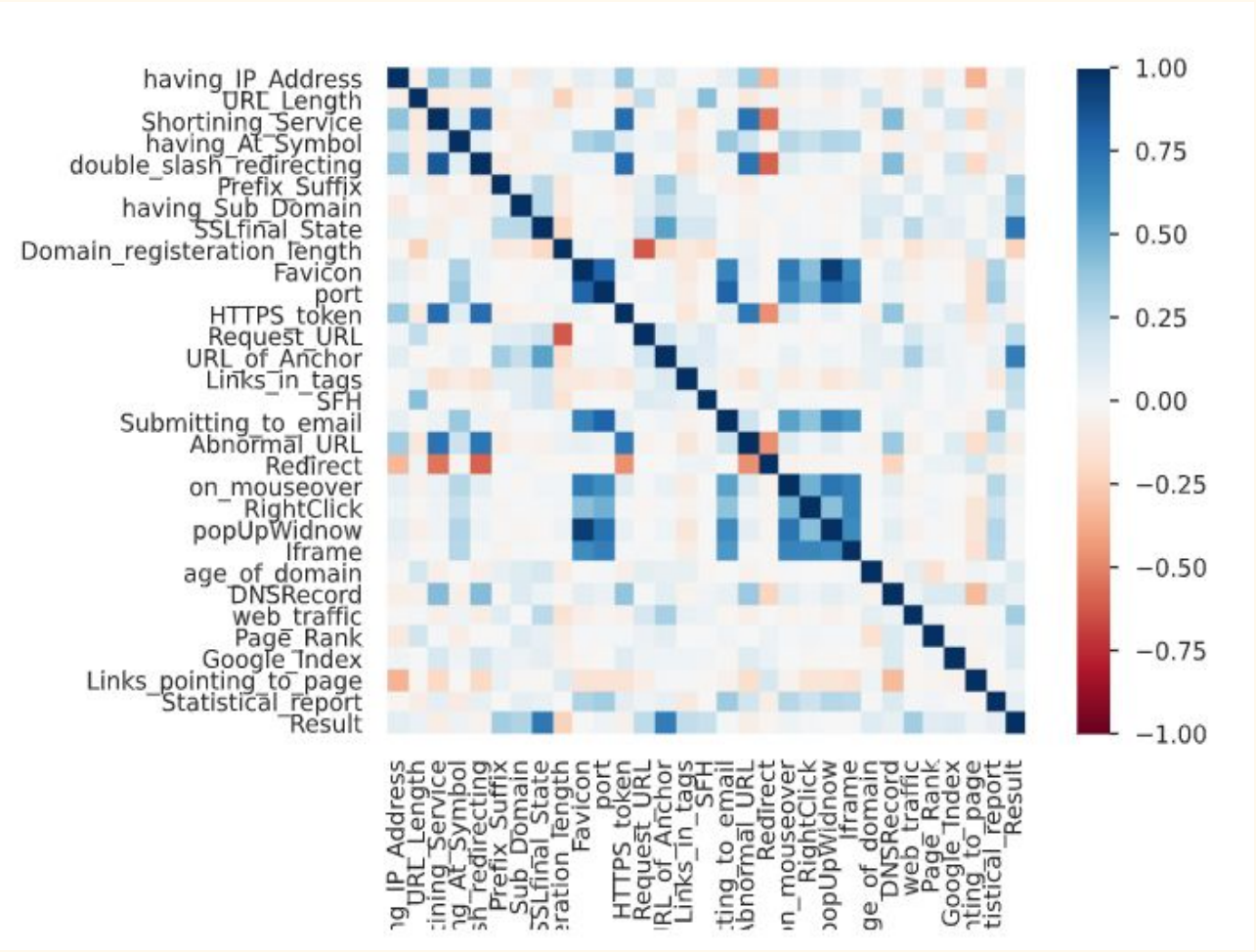
Distribution of the classes in the dataset: It reveals if the dataset is an imbalanced dataset or not and if sophisticated techniques like SMOTE, ADASYN etc can be incorporated to deal with the situation. The balance of the classes is not very much poor it seems.



Correlation Study: We used Pearson's correlation coefficient(r) which is a measure of the strength of the association between the two variables. This analysis is important to figure out the most relevant features with respect to the target variable.

$$r = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})/(n-1)}{\hat{\sigma}_X \hat{\sigma}_Y}.$$

We also obtained the top 10 and least 5 correlated variables which can be observed below:




```
Result                      1.000000
SSLfinal_State              0.714741
URL_of_Anchor               0.692935
Prefix_Suffix               0.348606
web_traffic                 0.346103
having_Sub_Domain           0.298323
Request_URL                 0.253372
Links_in_tags               0.248229
SFH                         0.221419
Google_Index                0.128950
Name: Result, dtype: float64

double_slash_redirecting      -0.038608
HTTPS_token                   -0.039854
Abnormal_URL                  -0.060488
Shortining_Service            -0.067966
Domain_registeration_length   -0.225789
Name: Result, dtype: float64
```

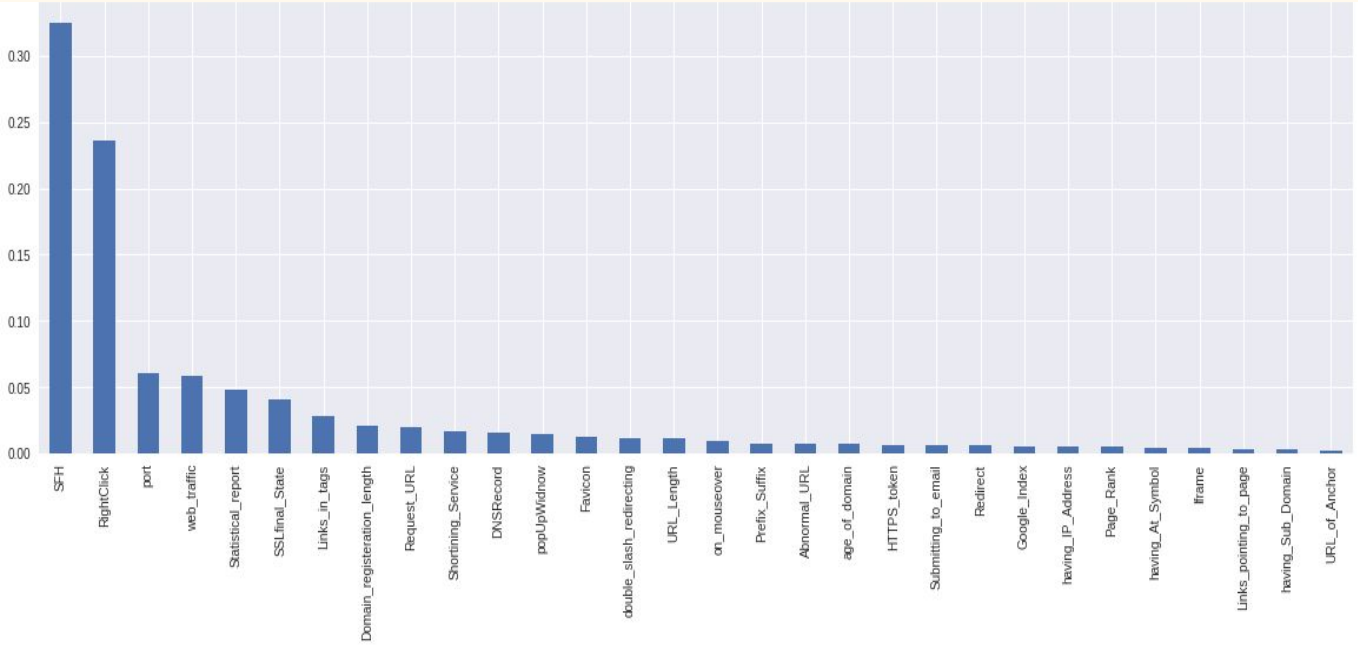
Bi-variate Analysis:

The t-test tells how significant the differences between groups are. Every t-value has a p-value to go with it. A p-value is the probability that the results from your sample data occurred by chance. They are usually written as a decimal. Low p-values are good as they indicate that data did not occur by chance. So, we calculated t-stats and p-value for each of the variables for better analysis.

Percentage of observations per label: There are 30 categorical variables as per the analysis. Histograms are obtained to analyze the count of each category corresponding to a particular label. This is shown below:

Outlier Analysis: Box plot study was done in order to observe the outliers present in each variable. After analysis, there were no outliers found.

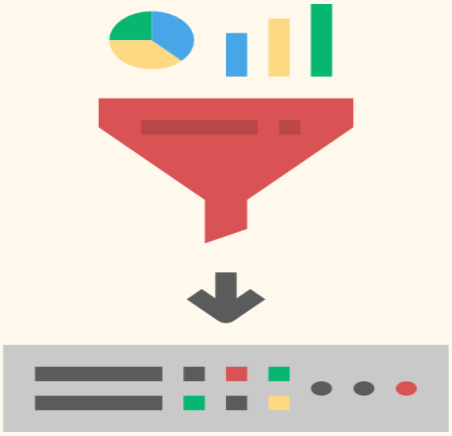
Feature importance: We derived the importance of each variable and ranked them from the most to least important as shown below:



STEP 4 (Pre-processing and Feature Selection):

As per the analysis obtained from the previous step, we did perform some preprocessing steps to filter our data.

There weren't any missing values. Then LabelEncoder was used to handle all the categorical variables. When we analysed the feature importances corresponding to the result variable, it was very difficult to distinguish between different variables as some of them shared almost equal feature importance. So, we came up with another feature selection technique most suitable for our problem.



Also, we obtained the data for duplicate rows as shown below and removed them from our dataset and also checked if there were any duplicate columns. Now, our dataset contains around 5849 rows.

Duplicate rows	5206
Duplicate rows (%)	47.1%

Feature Selection:

We employed the technique of feature addition for feature selection in which we initially calculated roc-auc score considering all variables and stored it. Then, we started adding one variable at a time from the most to least important and built an xgboost model at each round. If the increase between new roc-auc and previous roc-auc is more than a small threshold, we will keep the variable else we will discard it. This way, we filtered out the top 16 variables out of 30 which were most relevant to our problem domain. Those are:

```
['SFH', 'web_traffic', 'Statistical_report','SSLfinal_State',
'Links_in_tags', 'Domain_registeration_length','popUpWidnow',
 'Prefix_Suffix','double_slash_redirecting','URL_Length',
'on_mouseover', 'Abnormal_URL', 'HTTPS_token',
'having_At_Symbol', 'having_Sub_Domain', 'URL_of_Anchor']
```

STEP 5 (Model Building):

We performed comparison for our baseline model selection for which we took into consideration the following algorithms and compared their classification scores:

```
#Algorithm comparison
algorithms = {
        "DecisionTree": tree.DecisionTreeClassifier(max_depth=10),
        "RandomForest": ske.RandomForestClassifier(n_estimators=50),
        "GradientBoosting": ske.GradientBoostingClassifier(n_estimators=50),
        "AdaBoost": ske.AdaBoostClassifier(n_estimators=100),
        "GNB": GaussianNB()
    }

results = {}
print("\nNow testing algorithms")
for algo in algorithms:
    clf = algorithms[algo]
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    print("%s : %f %%" % (algo, score*100))
    results[algo] = score
```

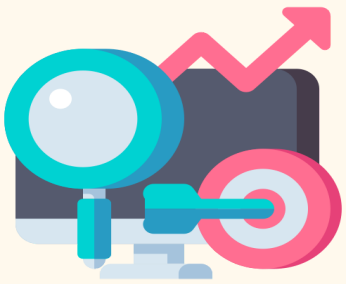
STEP 6 (Performance Evaluation):

```
Now testing algorithms
DecisionTree : 99.018472 %
RandomForest : 99.413256 %
GradientBoosting : 98.717856 %
AdaBoost : 98.594712 %
GNB : 70.336834 %

Winner algorithm is RandomForest with a 99.413256 % success
```

The winning algorithm was RandomForest with maximum classification score as shown. So, it was chosen as our model. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean/average prediction of the individual trees.

STEP 7 (Hyper-Parameter Tuning):



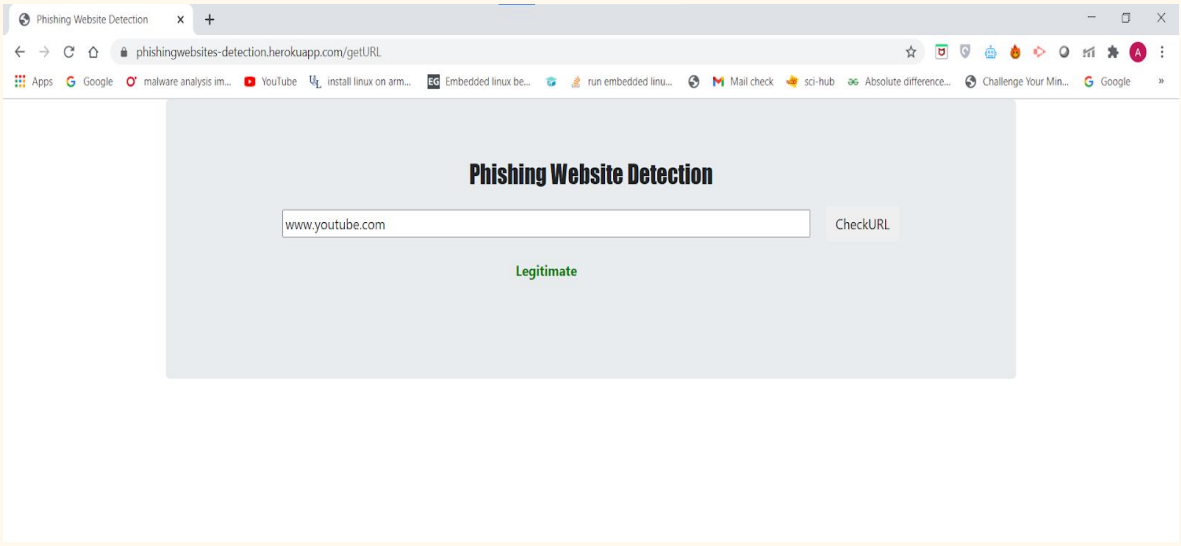
The main parameters used by a Random Forest Classifier are: criterion: the function used to evaluate the quality of a split, max_depth: maximum number of levels allowed in each tree, max_features: maximum number of features considered when splitting a node, min_samples_leaf: minimum number of samples which can be stored in a tree leaf, min_samples_split: minimum number of samples necessary in a node to cause node splitting and n_estimators: number of trees in the ensemble.

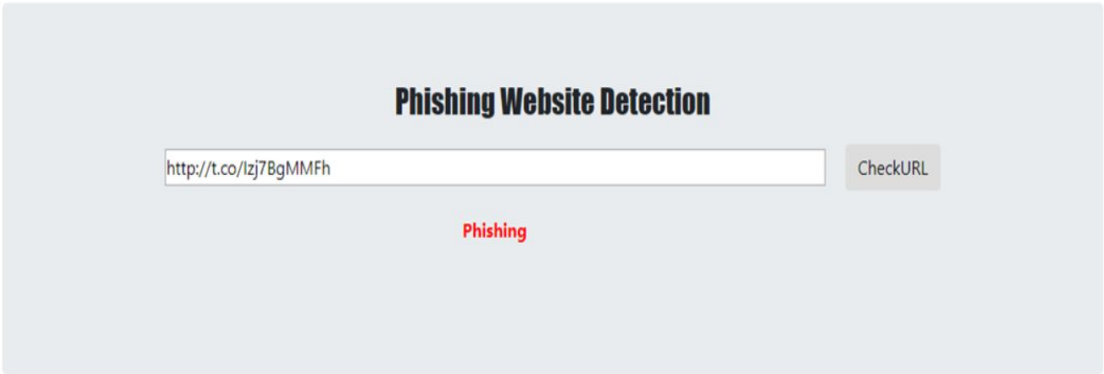
```
### Randomized Search Cv
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
max_features = ['auto', 'sqrt','log2']
max_depth = [int(x) for x in np.linspace(10, 1000,10)]
min_samples_split = [2, 5, 10,14]
min_samples_leaf = [1, 2, 4,6,8]
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
              'criterion':['entropy','gini']}

rf=RandomForestClassifier()
rf_randomcv=RandomizedSearchCV(estimator=rf,param_distributions=random_grid,n_iter=50,cv=2,verbose=2,
                               random_state=100,n_jobs=-1)
### fit the randomized model
rf_randomcv.fit(X_train,y_train)
```

```
Test selected features ROC AUC=0.985749
```

GUI/DEMO:





CONCLUSION

We successfully completed the detailed analysis on how an actual phishing websites detector works and operates. We presented a step by step procedure for this data mining project. In this report, we mentioned all our decisions and choices based on careful consideration of various factors relevant to our data. We reviewed various ways for exploratory data analysis, pre-processing, modelling as well as performance optimization methodologies. We also presented a user-friendly web application by which a user can easily detect a phishing website.

REFERENCES

Phishing Detection: Analysis of Visual Similarity Based Approaches

Detection of phishing websites using an efficient feature-based machine learning framework

Detecting Phishing Websites Using Machine Learning

Malicious web content detection using machine leaning - IEEE Conference Publication

(PDF) Swarm Intelligence Approaches for Parameter Setting of Deep Learning Neural Network: Case Study on Phishing Websites Classification

Detecting Phishing Websites using Neural Networks

Detecting phishing websites using machine learning | by Sayak Paul | Intel Software Innovators

Phishytics - Machine Learning for Detecting Phishing Websites - Securities

Detecting Phishing Websites using Machine Learning

(PDF) Predicting Phishing Websites based on Self-Structuring Neural Network

Phishing Websites Features

Phishing Websites Data Set

T Test (Student's T-Test): Definition and Examples